Military Surveillance Robot

Submitted in partial fulfillment of the requirements for the degree of

Bachelor of Technology

In

Computer Science and Engineering

by

Anagha Ramaswamy - 17BEC0152
Aishwarya Tapadiya - 17BEC0194
Dhruv Mittal - 17BCE2110

Under the guidance of

Prof. Sathiya Kumar
SCOPE VIT, Vellore.

Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

DECLARATION

I hereby declare that the thesis entitled "Military Surveillance Robot" submitted by me, for the award of the degree of *Bachelor of Technology in Electronics and Communication Engineering* to VIT is a record of bonafide work carried out by me under the supervision of Prof. Sathiya Kumar C.

I further declare that the work reported in this thesis has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university.

Place:Vellore

Date: 04/06/2020

CERTIFICATE

This is to certify that the thesis entitled "Military Surveillance Robot" submitted by **Anagha Ramaswamy(17BEC0152), Aishwarya Tapadiya(17BEC0194), Dhruv Mittal(17BCE2110)**, VIT, for the award of the degree of *Bachelor of Technology in Electronics and Communication Engineering*, is a record of bonafide work carried out by him / her under my supervision during the period, 01. 12. 2019 to 05.06.2020, as per the VIT code of academic and research ethics.

The contents of this report have not been submitted and will not be submitted either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university. The thesis fulfills the requirements and regulations of the University and in my opinion meets the necessary standards for submission.

Place : Vellore

Date : 05/06/2020

Signature of the Guide

Internal Examiner

External Examiner

Head of the Department

ACKNOWLEDGEMENT

We express our gratitude to Prof. Sathiya Kumar C. His support throughout the process kept us motivated to finish the project.

Apart from our team efforts, the completion of the project depends largely on the encouragement and guidelines of many others. We take this opportunity to express our gratitude to the people who have been instrumental in the successful completion of this project.

Further, We thank VIT for providing us this opportunity.

Anagha Ramaswamy
Aishwarya Tapadiya
Dhruv Mittal

EXECUTIVE SUMMARY

The thesis focuses on making standalone robots which can function in coordination of humans.

There are various types of command control robots in the market but the need of self-controlled, standalone robots is increasing in the military sector.

The various devices are connected by using the Internet of things so that the robot could be operated from any location. The robot has features like GPS location, monitoring the environment through visuals, obstacle detection. These properties can ultimately help a soldier, increase the survival rate and can also be used for investigational purposes.

The integration of the robot with motion detecting using the PIR sensor, obstacle detection using ultrasonic Sensor, tracking location using the GPS module and live monitoring using the camera module makes the robot useful for surveillance purposes.

CONTENT

INTRODUCTION

Objective:

We are focusing on making a standalone robot which can work in coordination with the humans. IoT will be used as a way of communication, so that we can operate the robot from any location. They can be operated through the LAN Network that is used by the soldiers in the military field.

The robot will have abilities like motion sensing, GPS location detection, monitoring the environment through visuals, obstacle detection. These properties can ultimately help a soldier, increase the survival rate and can also be used for investigational purposes. The ultrasonic sensor is a low-cost sensor which is used to detect and avoid the obstacles coming in the path of robot travel in a dynamic environment.

This project presents an approach for surveillance at remote and border areas using multifunctional robot based on current IOT used in defense and military applications. This robot has the ability to substitute the soldier at the border area to provide surveillance. This robot used to detect the presence of any enemy capture it in camera and give live streaming to the authorized person surveillance is the major role.

Motivation:

Surveillance is the major thing we need to secure anything as it is tedious job people are getting boarded because of that it might be risky to observe all things we are going to make a robot which continuously monitors everything. Because of that monitoring the work will be somewhat easy and more accurate because of technology.

There is a huge need for self-controlled robots for military and security purposes, these robots can be of great use to the soldiers on the battle ground. It can create an alarm if there’s danger and the risk of losing soldiers on the field is reduced. They are designed to standalone and work efficiently and in coordination with humans. With the help of robots, we are reducing the casualties automatically.

The implementation of this project to resolve the problem of replacing humans with surveillance robots, because of this we reduce harm to human resources. Robots are usually miniature in size so they are capable of entering tunnels, mines and small holes in buildings and also have capability to survive in harsh and difficult climatic conditions for a long time without causing any harm.

Background:

There are very few bots that are designed specifically for military use. After the concept of a robot that can efficiently perform an archeological survey came about, the need for one such robot in the military field is seen. The devices available nowadays are not very responsive and fast as they need to be. We are combining archeological surveying with specifications needed for the military field to design our robot. The robot will have features of IoT for a faster and more responsive working of the device.

The military is undeniably the primary customer of new advances and improvements in strategy, and is also often the sponsor of new improvements when it comes to envisioning new innovations in military settings. Numerous basic military technologies deployed out of the blue are now advanced to the piece of industrial robots.

PROJECT DESCRIPTION AND GOALS

To establish an idea of building prototype self- controlled robots for border patrol, surveillance and in active combat both as a standalone unit, it is maneuvered in automatic mode and manual. The robot is a machine capable of performing tasks autonomously or semi-autonomously which is guided by the micro-controllers.

Many industries are using robots to perform the task with a higher degree of precision. Accordingly, in light of the requirement and purpose of usage the task of sensor integration is achieved. In this research work the proposed robot is capable of detecting and avoiding obstacles in the travel path of the robot under dynamic environments.

We will be using many IOT devices such as GPS Module, PIR Sensor for motion sensing, Arduino Camera Module for monitoring, Ultrasonic sensor for obstacle detection.

Our goal is to make a robot which is capable of doing all tasks mentioned and provide better surveillance for the military than ever before which makes the border more secure not only for the military soldier at the border but also keeps the citizens of the nation safe.

TECHNICAL SPECIFICATION

IoT can be conceivable in every field and also provides appropriate solutions for a wide range of applications like smart towns, traffic congestion controlling systems, waste management services, structural health services, security & emergency services, logistics, retails, industrial services.

It uses GPS, magnetic compass and adjusts strategies based on surroundings using path planning and obstacle detection algorithms. GPS navigation methodology is acquired and implemented gives a clear idea of establishing and pairing an interface of GPS using Node MCU.

Automating concepts has been clearly studied and implemented from this. It consists of information on how to program a microcontroller using an Ultrasonic sensor and sense the motion via PIR sensors.

To create a robot that can be used as a spy bot for security purposes, we will be using teleoperations to get the GPS coordinates and then move the robot in the specified direction. Using IOT, we will send the responses with less delay.

Users can see live streaming from a computer device as a website or phone application as the camera is attached. Different buttons are there such as Forward, Reverse, Left, Right and Stop to control the Robot.

DESIGN APPROACH AND DETAILS

Design Approach - Flowchart:

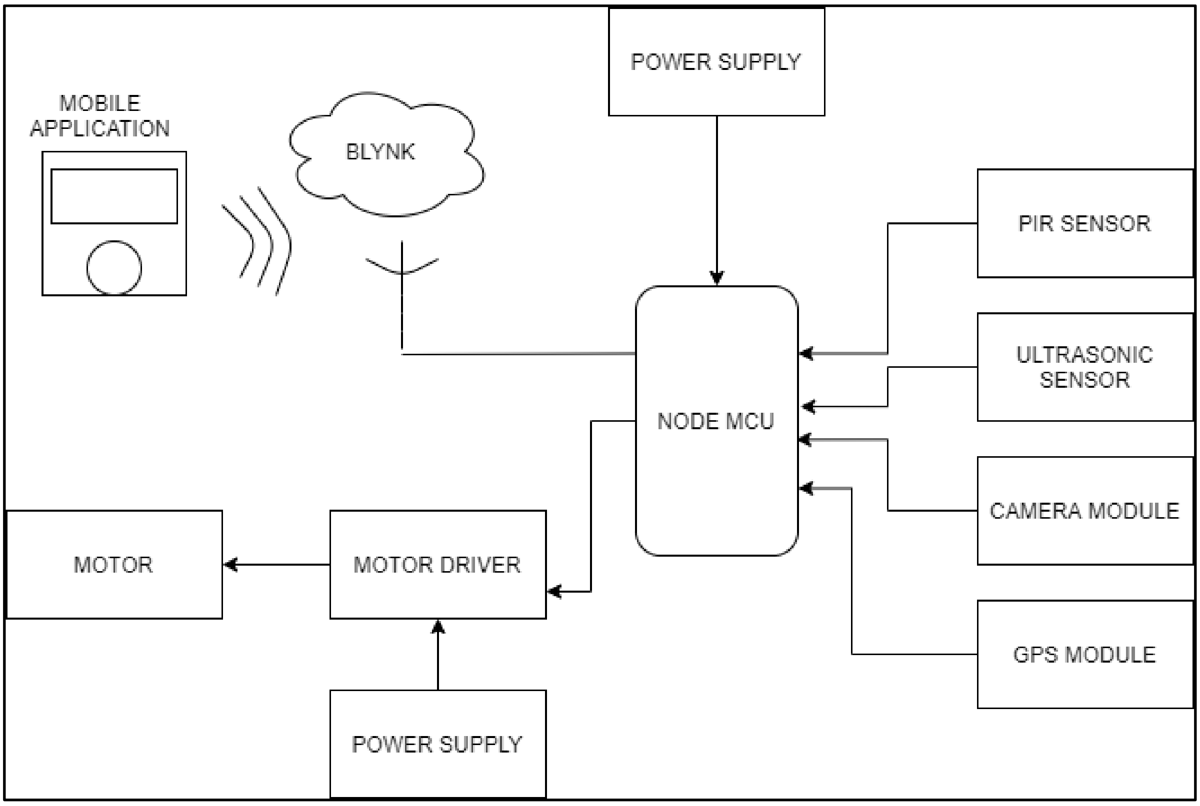


Fig: Flowchart of the design

Components:

1. PIR Sensor:
   Passive Infrared sensors allow us to sense motion.
   The sensor has 2 slots. When it is idle, both the slots detect the same amount of IR. When a warm body passes by, it first intercepts one half of the sensor which causes a positive differential change between the two halves. The reverse happens when a body leaves the area. The change in pulse is detected and output is produced.

Fig: PIR Sensor

2. GPS Module
   The Ublox Neo 6M is a GPS receiver. It has a data backup battery which saves the data incase of accidental power shut downs. The location of out Tobot will be tracked using the module. The robot will be given the coordinates of a specific location and it will guide itself there.

Fig: GPS Module

3. Node MCU
   NodeMCU is an open-source firmware and development kit that helps us to prototype or build IoT products. It includes firmware that runs on the ESP8266 Wi-Fi SoC from Espressif Systems, and hardware which is based on the ESP-12 module. The firmware uses the Lua scripting language. It is based on the eLua project and built on the Espressif Non-OS SDK for ESP8266.

   We will be using NodeMCU to connect the sensors and the modules. The data collected will be transferred via WiFi to Blynk that is connected to a mobile application.

Fig: NodeMCU

4. Arduino camera module
   This is an Arduino camera module, adopted the Surveillance cameras digital image processing chip-OV0706, specially designed for image acquisition and processing application, based on TTL communication interface, very convenient to connect with Arduino controller, able to read image and data via UART serial port, and then perform some image processing.

Fig: Arduino Camera Module

5. Motor Driver
   The L293D is a 16-Pin Motor Driver. It is mainly used to drive motors. A single L293D Motor Driver is capable of running two DC motors at the same time with the direction of these two motors that can be controlled independently. This will be used to drive the motors of the device.

Fig: Motor Driver L293D

6. Motors
   A DC motor is any of a class of rotary electrical motors that converts direct current electrical energy into mechanical energy. The most common types rely on the forces produced by magnetic fields. Nearly all types of DC motors have some internal mechanism, either electromechanical or electronic, to periodically change the direction of current in part of the motor.

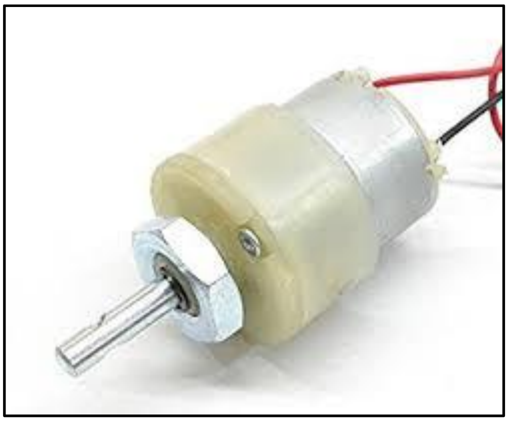


Fig: DC Motor

7. Magnetometer
   The HMC5883L helps the robot to be aware about the direction it is pointing towards. The GPS Module gets the direction sense only after the robot has travelled in a certain direction for a certain amount of time, therefore it is important to use a magnetometer.

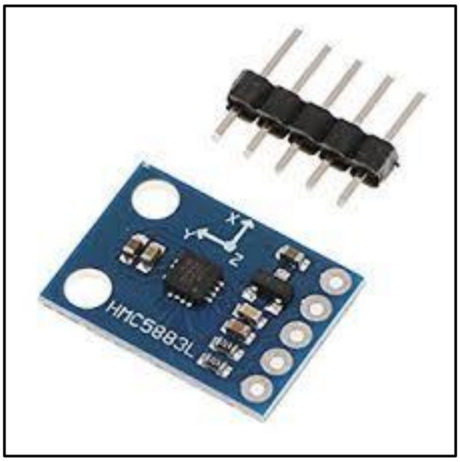


Fig: Magnetometer

8. Ultrasonic Sensor
   The sensor will be used to detect the obstacles in the path of the Robot. This sensor works with a transmitter and receiver mechanism which transmits ultrasonic waves ahead of the robot. If an obstacle is detected the waves are reflected back and received by the ultrasonic sensor which signals to the robot about the obstacle ahead.

Fig: Ultrasonic Sensor

Codes and Standards:

```
#include <ESP8266WiFi.h>
#include <WiFiClient.h>
#include <BlynkSimpleEsp8266.h>
#include <ESP8266WebServer.h>
#include <Wire.h>
#include <ArduCAM.h>
#include <SPI.h>
#include "memorysaver.h"
#include "Wire.h"
#include "HMC5883L.h"

HMC5883L compass;

const int pirPin = 2;
const int trigPin = 9;
const int echoPin = 10;

int EN1 = D0;
int EN2 = D1;
const int motorIn1 = D6;
const int motorIn2 = D7;
const int motorIn3 = D4;
const int motorIn4 = D3;

const int CS = 16;
static const size_t bufferSize = 4096;
static const int fileSpaceOffset = 700000;
static const int wifiType = 0;
const char *AP_ssid = "arducam_esp8266";

const char auth [] = "YourAuthToken";
const char *ssid = "yourWiFiSSID";
const char *password = "yourWiFiPassword";

static IPAddress ip(192, 168, 1, 203);
static IPAddress gateway(192, 168, 1, 1);
static IPAddress subnet(255, 255, 255, 0);

const String fName = "res.txt";

int fileTotalKB = 0;
int fileUsedKB = 0; int fileCount = 0;
String errMsg = "";
int imgMode = 1; // 0: stream  1: capture
int resolution = 3;

ESP8266WebServer server(80);

ArduCAM myCAM(OV2640, CS);

int calibrationTime = 30;
long unsigned int lowIn;
long unsigned int pause = 5000;
boolean lockLow = true;
boolean takeLowTime;

int PIRValue = 0;
long duration;
int distance;

void setup()
{
   Serial.begin(115200);
   pinMode(pirPin, INPUT);
   pinMode(trigPin, OUTPUT);
   pinMode(echoPin, INPUT);

   pinMode(motorIn1, OUTPUT);
   pinMode(motorIn2, OUTPUT);
   pinMode(motorIn3, OUTPUT);
   pinMode(motorIn4, OUTPUT);
   pinMode(EN1, OUTPUT);
   pinMode(EN2, OUTPUT);

   analogWrite(EN1, 0);
   analogWrite(EN2, 0);

  uint8_t vid, pid;
  uint8_t temp;

  Serial.println("ArduCAM Start!");
  pinMode(CS, OUTPUT);
  SPI.begin();
  SPI.setFrequency(4000000); //4MHz

  myCAM.InitCAM();

    WiFi.mode(WIFI_STA);
    WiFi.begin(ssid, password);
    while (WiFi.status() != WL_CONNECTED) {
      delay(500);
      Serial.print(".");
    }
    WiFi.config(ip, gateway, subnet);
    Serial.println("WiFi connected");
    Serial.println("");
    Serial.print("ip: ");
    Serial.println(WiFi.localIP());

  // setup handlers
  server.on("/capture", HTTP_GET,
serverCapture);
  server.on("/stream", HTTP_GET, serverStream);
  server.on("/submit", handleSubmit);
  server.on("/clear", clearData);
  server.onNotFound(handleNotFound);
  server.begin();
  Serial.println("Server started");

  Blynk.begin(auth, ssid, password);
}

void loop()
{
   PIRSensor();
   UltrasonicSensor();
   CameraSensor();
   GpsCoordinates();
}

void PIRSensor()
{
   if(digitalRead(pirPin) == HIGH) {
      if(lockLow) {
         PIRValue = 1;
         Blynk.virtualWrite(V0, PIRValue);
         lockLow = false;
         Serial.println("Motion detected.");
         delay(50);
      }
      takeLowTime = true;
   }
   if(digitalRead(pirPin) == LOW) {
      if(takeLowTime){
         lowIn = millis();takeLowTime = false;
      }
      if(!lockLow && millis() - lowIn > pause) {
         PIRValue = 0;
         Blynk.virtualWrite(V0, PIRValue);
         lockLow = true;
         Serial.println("Motion ended.");
         delay(50);
      }
   }
}
void UltrasonicSensor()
{
digitalWrite(trigPin, LOW);
delayMicroseconds(2);
digitalWrite(trigPin, HIGH);
delayMicroseconds(10);
digitalWrite(trigPin, LOW);
duration = pulseIn(echoPin, HIGH);
distance= duration*0.034/2;
Blynk.virtualWrite(V1, distance);
if(distance<=5)
{
  Serial.println("Obstacle Detected!");
}
}

void CameraSensor()
{
  server.handleClient();
}
void GpsCoordinates()
{
  getGPS();
  setWaypoint();
  getCompass();
  gpsInfo();
}

void start_capture()
{
  myCAM.clear_fifo_flag();
  myCAM.start_capture();
}
```

```
void camCapture(ArduCAM myCAM) {
  WiFiClient client = server.client();

  size_t len = myCAM.read_fifo_length();
  if (len >= 0x07ffff) {
    Serial.println("Over size.");
    return;
  } else if (len == 0 ) {
    Serial.println("Size is 0.");
    return;
  }
  myCAM.CS_LOW();
#if !(defined (ARDUCAM_SHIELD_V2) &&
defined (OV2640_CAM))
  SPI.transfer(0xFF);
#endif

  if (!client.connected()) return;
  static uint8_t buffer[bufferSize] = {0xFF};
  while (len) {
    size_t will_copy = (len < bufferSize) ? len :
bufferSize;
    SPI.transferBytes(&buffer[0], &buffer[0],
will_copy);
    if (!client.connected()) break;
    client.write(&buffer[0], will_copy);
    len -= will_copy;
  }

  myCAM.CS_HIGH();
}
void serverCapture() {

  start_capture();
  Serial.println("CAM Capturing");

  int total_time = 0;

  total_time = millis();
  while (!myCAM.get_bit(ARDUCHIP_TRIG,
CAP_DONE_MASK));
  total_time = millis() - total_time;
  Serial.print("capture total_time used (in
miliseconds):");
  Serial.println(total_time, DEC);

  total_time = 0;

  Serial.println("CAM Capture Done!");
  total_time = millis();
  camCapture(myCAM);
  total_time = millis() - total_time;
  Serial.print("send total_time used (in
miliseconds):");
  Serial.println(total_time, DEC);
  Serial.println("CAM send Done!");
}
void serverStream() {
  WiFiClient client = server.client();

  String response = "HTTP/1.1 200 OK\r\n";
  response += "Content-Type: multipart/x-mixed-
replace; boundary=frame\r\n\r\n";
  server.sendContent(response);

  while (client.connected()) {
    start_capture();
    while (!myCAM.get_bit(ARDUCHIP_TRIG,
CAP_DONE_MASK));
    size_t len = myCAM.read_fifo_length();
    if (len >= 0x07ffff) {
      Serial.println("Over size.");
      continue;
    } else if (len == 0 ) {
      Serial.println("Size is 0.");
      continue;
    }

    myCAM.CS_LOW();
    myCAM.set_fifo_burst();
#if !(defined (ARDUCAM_SHIELD_V2) &&
defined (OV2640_CAM))
    SPI.transfer(0xFF);
#endif
   if (!client.connected()) break;
    response = "--frame\r\n";
    response += "Content-Type:
image/jpeg\r\n\r\n";
    server.sendContent(response);

    static uint8_t buffer[bufferSize] = {0xFF};
```

```
 while (len) {
      size_t will_copy = (len < bufferSize) ? len :
bufferSize;
      SPI.transferBytes(&buffer[0], &buffer[0],
will_copy);
      if (!client.connected()) break;
      client.write(&buffer[0], will_copy);
      len -= will_copy;
    }
    myCAM.CS_HIGH();

    if (!client.connected()) break;
  }
}
void handleNotFound() {

  String ipStr = WiFi.localIP().toString();
  Serial.print("server uri: " );
Serial.println(server.uri());
  if (server.uri().indexOf(".jpg") != -1 )
  {
    loadFromSpiffs(server.uri());
    return;
  }

  //// default HTML
  String message = "<html><head>\n";
  message += "</head><body>\n";
  message += "<form action=\"http://" + ipStr +
"/submit\" method=\"POST\">";
  message += "<h1>ESP8266-12E Arducam
Mini</h1>\n";
  if (errMsg != "")
    message += "<div style=\"color:red\">" +
errMsg + "</div>";

  if (imgMode == 0) // stream mode
  {
    message += "<div><h2>Video
Streaming</h2></div> \n";
    message += "<div><img id=\"ArduCam\"
src=\"http://" + ipStr + "/stream\" ></div>\n";
    imgMode = 1; // set back to capture mode so it
doesn't get stuck in streaming mode

  }
  else
  {
    message += "<div>";
    message += getDropDown();

    message += " <input type=\"radio\" id=\"strm\"
name=\"imgMode\" value=\"stream\" ";

    if (imgMode == 0)
      message += " checked ";
    message += "> Stream \n";

    message += " <input type=\"radio\" id=\"capt\"
name=\"imgMode\" value=\"capture\"  ";
    if (imgMode == 1)
      message += " checked ";
    message += "> Capture \n";

    message += "  <input type='submit'
value='Submit'  >\n";
    message += "    <a style=\"font-
size:12px; font-weight:bold;\" href=\"http://" +
ipStr + "\">Refresh</a>";
    message += "     <a style=\"font-
size:12px; font-weight:bold;\" onclick=\"return
confirm('Are you sure? This will delete all stored
images.')\" ";
    message += " href=\"http://" + ipStr +
"/clear\">Clear Data</a>\n";
    message += "</div>\n";
    fileTotalKB = (int)fs_info.totalBytes;
    fileUsedKB = (int)fs_info.usedBytes;

    if (fileCount > 0)
    {
      int percentUsed = ((float)fileUsedKB /
(float)(fileTotalKB - fileSpaceOffset)) * 100;
      String colorStr = "green";
      if (percentUsed > 90)
        colorStr = "red";

      message += "<div style=\"width:450px;
background-color:darkgray; padding:1px;\">";
      message += "<div style=\"position:absolute;
color:white; padding-top:2px; font-size:11px;\">
  space used: " + String(percentUsed) +
"%</div>";
      message += "<div style=\"width:" +
String(percentUsed) + "%; height:16px;
background-color: " + colorStr +
";\"></div></div>\n";

    }
```

```
    message += "<table><tr>";
    int colCnt = 0;
    for (int i = 1; i <= fileCount; i++)
    {
      message += "<td><a href=\"/pics/" + String(i)
+ ".jpg\">" + i + ".jpg</a></td>\n";

      colCnt++;
      if (colCnt >= 10) //  columns
      {
        message += "</tr><tr>";
        colCnt = 0;
      }
    }
    message += "</tr></table>\n";
    message += "<div><img id=\"ArduCam\"
src=\"http://" + ipStr + "/capture\" ></div>\n";
  }

  message += "</form> \n";
  message += "</body></html>\n";

  server.send(200, "text/html", message);
}


void getGPS()                                    // Get
Latest GPS coordinates
{
    while (Serial2.available() > 0)
    gps.encode(Serial2.read());
}


void setWaypoint()                               //
Set up to 5 GPS waypoints
{
if (wpCount >= 0)
  {
    Serial1.print("GPS Waypoint ");
    Serial1.print(wpCount + 1);
    Serial1.print(" Set ");
    getGPS();                                    // get
the latest GPS coordinates
    getCompass();                                //
update latest compass heading

    Home_LATarray[ac] = gps.location.lat();
// store waypoint in an array
    Home_LONarray[ac] = gps.location.lng();
// store waypoint in an array

    Serial.print("Waypoint #1: ");
    Serial.print(Home_LATarray[0],6);
    Serial.print(" ");
    Serial.println(Home_LONarray[0],6);
    Serial.print("Waypoint #2: ");
    Serial.print(Home_LATarray[1],6);
    Serial.print(" ");
    Serial.println(Home_LONarray[1],6);
    Serial.print("Waypoint #3: ");
    Serial.print(Home_LATarray[2],6);
    Serial.print(" ");
    Serial.println(Home_LONarray[2],6);
    Serial.print("Waypoint #4: ");
    Serial.print(Home_LATarray[3],6);
    Serial.print(" ");
    Serial.println(Home_LONarray[3],6);
    Serial.print("Waypoint #5: ");
    Serial.print(Home_LATarray[4],6);
    Serial.print(" ");
    Serial.println(Home_LONarray[4],6);

    wpCount++;                                   //
increment waypoint counter
    ac++;                                        //
increment array counter

  }
  else {Serial1.print("Waypoints Full");}
}


void getCompass()                                //
get latest compass value
 {

  Vector norm = compass.readNormalize();

  // Calculate heading
  float heading = atan2(norm.YAxis, norm.XAxis);
  if(heading < 0)
    heading += 2 * M_PI;
  compass_heading = (int)(heading * 180/M_PI);
// assign compass calculation to variable
(compass_heading) and convert to integer to
remove decimal places

 }


void gpsInfo()                                   //
displays Satellite data to user
  {
        Number_of_SATS =
(int)(gps.satellites.value());         //Query Tiny
GPS for the number of Satellites Acquired
        Distance_To_Home =
TinyGPSPlus::distanceBetween(gps.location.lat(),
gps.location.lng(),Home_LATarray[ac],
Home_LONarray[ac]);  //Query Tiny GPS for
Distance to Destination
        Serial1.print("Lat:");
        Serial1.print(gps.location.lat(),6);
        lcd.print(0,0, gps.location.lat());

        Serial1.print(" Lon:");
        Serial1.print(gps.location.lng(),6);
        lcd.print(0,1, gps.location.lng());
        Serial1.print(" ");

        Serial1.print(" SATs ");
        Serial1.print(Number_of_SATS);

        Serial.print("Distance to Home ");
        Serial.println(Distance_To_Home);
        Serial1.print("m");
  }


BLYNK_WRITE(V2)
{
  int x = param[0].asInt();
  int y = param[1].asInt();
}
void navigation()
{

  speed = Blynk.virtualRead(V3);
  analogWrite(EN1, speed);
  analogWrite(EN2, speed);

  if(x==1 && y==2)
  {
    digitalWrite(motorIn1, HIGH);
    digitalWrite(motorIn2, LOW);
    digitalWrite(motorIn3, HIGH);
    digitalWrite(motorIn4, LOW);
  }
  else if(x==1 && y==0)
  {
    digitalWrite(motorIn1, LOW);
    digitalWrite(motorIn2, HIGH);
    digitalWrite(motorIn3, LOW);
    digitalWrite(motorIn4, HIGH);
  }
  else if(x==2 && y==1)
  {
    digitalWrite(motorIn1, HIGH);
    digitalWrite(motorIn2, LOW);
    digitalWrite(motorIn3, LOW);
    digitalWrite(motorIn4, HIGH);
  }
  else if(x==0 && y==1)
  {
    digitalWrite(motorIn1, LOW);
    digitalWrite(motorIn2, HIGH);
    digitalWrite(motorIn3, HIGH);
    digitalWrite(motorIn4, LOW);
  }
  else
  {
    analogWrite(EN1, 0);
    analogWrite(EN2, 0);
  }
```

Constraints, Alternatives and Tradeoff:

The device is designed specifically for military purposes. The device should be connected to the local LAN network of the military so that it is allowed to use the internet. The device can further be modified for archeological purposes as well. The device will not be able to function properly if it is not connected to WiFi. Another option could be to use a GSM module for connectivity.

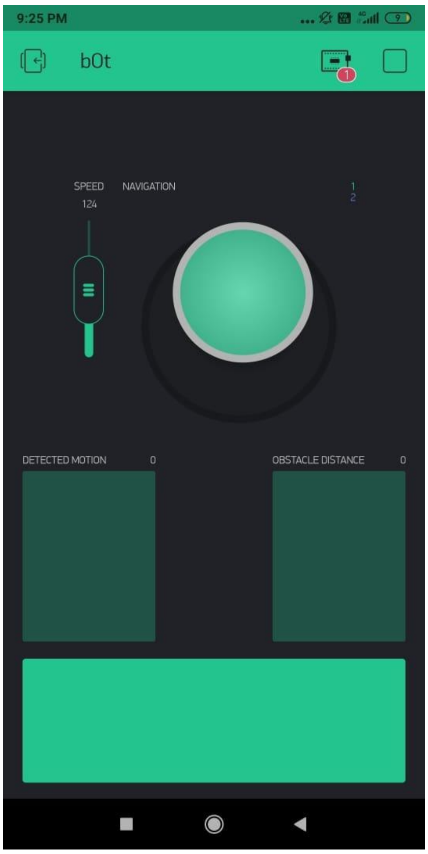
With a larger budget, modules to record voices can also be installed in the device. The autonomous device can also have the option to be driven manually as well. The size of the device is to be designed as small as possible so that it can enter areas that will be difficult for humans.

SCHEDULE, TASKS AND MILESTONES

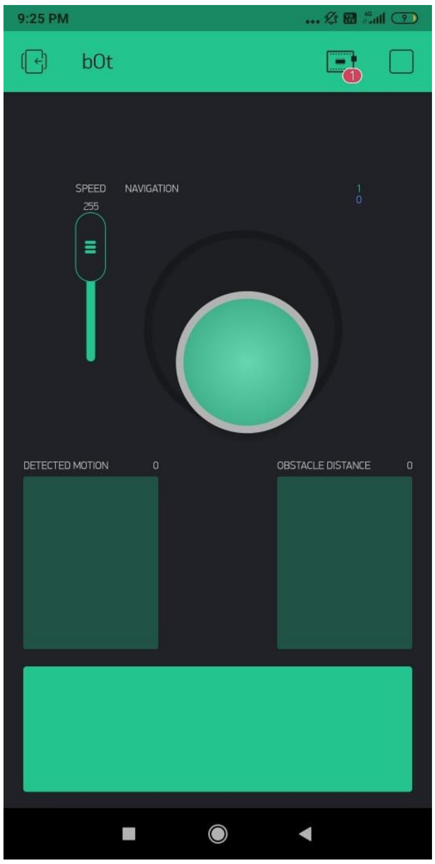
Selection of Problem Statement :	1st January 2020
Review 0 :	5th January 2020
Deciding the architecture:	10 to 15 January 2020
Review 1 :	6th February 2020
Finalising the Architecture:	27th February 2020
Testing of the components:	12th-15th March 2020
Writing Component Specific Codes:	5th May - 1 June 2020
Report Writing:	2nd - 4th June 2020

PROJECT DEMOSTRATION

Interface of the Blynk app.



Forward at almost 50% speed



Backward at 100% speed

RESULT AND DISCUSSION

The demonstration of the project is shown using a Blynk application. The outputs given by the sensors and GPS are shown on the app. The bot can perform both teleoperation and can be moved by the user. Controlling the bot can be done through the app. This will be suited to operate as a military bot.

Due to the pandemic situation, hardware demonstration of the robot was not possible. The code and mobile application were written and made as detailed as possible.

SUMMARY

The proposed robot is competent to take a decision and face situations with no human communication. The integration of the robot with motion detecting using the PIR sensor, obstacle detection using ultrasonic Sensor, tracking location using the GPS module and live monitoring using the camera module makes the robot useful for surveillance purposes.

It recognizes hindrances precisely and takes the choice to move to left, right, forward and in reverse according to the accessibility of adequate space. The camera module is helpful to consistently catch the pictures

The proposed research can be considered in future to improve the usage of self-routed robots with different sensors also.

REFERENCES

1. “Design of Low-Cost Self-Navigation Rover Based on IOT” by Santosh D Pandure and Pravin L Yannawar

2. “MULTIPLE MOTION CONTROL SYSTEM OF A ROVER BASED ON IOT TO PRODUCE CLOUD SERVICE” by MaradaniKeerthikanth , P.R.SaiSanath and Mr.S.Manikandaswamy.

3. “An Ultrasonic Sensor for Distance Measurement in Automotive Applications.” By Carullo A, Parvis M.

4. “Obstacle Avoidance Robot. International Journal of Science, Engineering and Technology Research” by Bhagat K, Deshmukh S.

5. “A sensor fusion methodology for obstacle avoidance robot. Electro Information Technology” by Aman MS, Mahmud MA, Jiang H, Abdelgawad A, Yelamarthi